# LAPORAN PRAKTIKUM
# PEMROGRAMAN BERORIENTASI OBJEK

## MODUL VI

## POLIMORFISME


Oleh:

Ramdhan Wijaya

2211102208

IF-10-M


**S1 TEKNIK INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**2024**

# I. DASAR TEORI

Polimorfisme adalah salah satu konsep penting dalam pemrograman berorientasi objek yang memungkinkan objek untuk mengambil bentuk atau perilaku yang berbeda dalam konteks yang berbeda. Ini memungkinkan kelas untuk memiliki banyak bentuk (polymorphic) dengan menggunakan pewarisan (inheritance) dan pengikatan dinamis

Redefinisi tunggal merujuk pada kemampuan kelas turunan untuk mengganti (override) implementasi metode yang diwarisi dari kelas induk (superclass). Dengan kata lain, kelas anak dapat memiliki versi yang berbeda dari metode yang didefinisikan di kelas induk, yang memungkinkan perilaku yang berbeda untuk diperoleh oleh objek yang berbeda dari kelas tersebut.

Fungsi virtual adalah metode yang dideklarasikan dalam kelas induk dengan kata kunci virtual, yang kemudian dapat di-redefinisikan di kelas anak dengan kata kunci override. Ini memungkinkan metode yang ditetapkan dalam waktu kompilasi (compile-time) untuk dipilih pada waktu eksekusi (runtime) berdasarkan jenis objek aktual yang digunakan.

Fungsi virtual murni (pure virtual function) adalah metode dalam kelas abstrak yang tidak memiliki implementasi. Metode ini dideklarasikan dalam kelas induk dan harus di-redefinisikan dalam kelas anak. Tujuannya adalah untuk memaksa kelas anak untuk menyediakan implementasi metode tersebut.

Abstract class adalah kelas yang tidak dapat diinstansiasi langsung, tetapi dapat digunakan sebagai kerangka dasar untuk kelas turunannya. Abstract class dapat memiliki metode abstrak (yang tidak memiliki implementasi) serta metode konkret (dengan implementasi). Untuk menggunakan abstract class, kelas turunannya harus menerapkan semua metode abstrak yang dideklarasikan dalam kelas induknya.

Abstract method adalah metode dalam abstract class yang dideklarasikan tanpa implementasi. Metode ini tidak memiliki tubuh metode (body) dan hanya memiliki deklarasi. Abstract method harus di-redefinisikan (diimplementasikan) dalam kelas turunannya. Jika sebuah kelas memiliki setidaknya satu metode abstrak, maka kelas tersebut harus dideklarasikan sebagai abstract class.

## II. GUIDED

### DEMO1.cpp

Source code

```
//ramadhan wijaya
//2211102208

#include <iostream>
#include <conio.h>
using namespace std;

class makhluk{
    public:
    virtual void keterangan(){  // deklarasi rposedur
virtual
        cout <<"keterangan() pada kelas mahkluk"<<endl;
    }
};

class mamalia:public makhluk{ // ":" mewariskan
    public:
    void keterangan(){
        cout<<"keterangan() pada kelas mamalia"<<endl;
    }
};

class sapi:public mamalia{
    void keterangan(){
        cout<<"keterangan() pada kelas sapi"<<endl;
    }
};

int main (){
    //definisi obek mamalia
    mamalia m,m2;

    //definisi objek sapii
    sapi s;

    //definisi pointer ke objek class mahkluk
    makhluk *binatang;

    //menunjuk ke objek mamalia
    //harus meunjuk ke alamatnya tidak bisa ke objeknya
langsung :::menggunakan "&"
```

```
    binatang = &m;
    binatang->keterangan();

    cout <<"--------------------------------"<<endl;
    //menuju ke objek sapi
    //harus meunjuk ke alamatnya tidak bisa ke objeknya
langsung :::menggunakan "&"
    binatang = &s;
    binatang->keterangan();

    cout <<"--------------------------------"<<endl;
    binatang = &m2;
    binatang->keterangan();
    getch();

    return 0;
}
```

Output :

```
keterangan() pada kelas mamalia

--------------------------------
keterangan() pada kelas sapi

--------------------------------
keterangan() pada kelas mamalia
```

Keterangan :

Program yang menggunakan fungsi virtual di class parent dan di panggil oleh class anak anaknya, alhasil class class childnya mempunyai fungsi yang sama yaitu fungsi keterangan();.

**DEMO2.cpp**

Source code

```
//ramadhan wijaya
//2211102208
#include <iostream>
#include <cstring>
```

```
using namespace std;

class Keluarga
{
public:
    char nama[20];

public:
    // konstruktor Keluarga
    Keluarga(const char *Nama)
    {
        strncpy(nama, Nama, sizeof(nama));
    }
    // destruktor Keluarga
    virtual ~Keluarga()
    {
        cout << "Destruktor di kelas Keluarga.." << endl;
    }
    // deklarasi fungsi virtual murni
    virtual void info() = 0;
};

class Keturunan : public Keluarga
{
private:
    char nama_depan[15];

public:
    Keturunan(const char *Nama_depan, const char *Nama_kel)
: Keluarga(Nama_kel)
    {
        strncpy(nama_depan, Nama_depan, sizeof(nama_depan));
    }
    ~Keturunan()
    {
        cout << "Destruktor di Keturunan.." << endl;
    }
    void info() override
    {
        cout << nama_depan << ' ' << nama << endl;
    }
};

int main()
{
```

```
    Keluarga *anak1 = new Keturunan("Umar", "Khatab");
    anak1->info();
    Keluarga *anak2 = new Keturunan("Udin", "Pambudi");
    anak2->info();

    delete anak1;
    delete anak2;

    cin.get(); // Tunggu masukan sebelum program selesai
    return 0;
}
```

Output :

```
Umar Khatab
Udin Pambudi
Destruktor di Keturunan..
Destruktor di kelas Keluarga..
Destruktor di Keturunan..
Destruktor di kelas Keluarga..
```

Keterangan :

sama seperti pada DEMO1.cpp tapi bedanya pada penggunaan fungsi virtual ditambahkan dengan override diakhir fungsi. Penulisan override diakhir tidaklah wajib namun sangat di anjurkan untuk memberitahu ke kompilator bahwa fungsi tersebut adalah fungsi dari class lain.

**Guided 1**

Source code

Anjing.java

```
/**
 *
 * @author rama
 * 2211102208
 */
public class Anjing extends Binatang {

    private final String nama;
```

```
    Anjing(String nama) {
        super("Anjing");
        this.nama = nama;
    }

    @Override
    public void suara() {
        System.out.println("menggonggong");
    }

    @Override
    public String toString() {
        return super.toString() + " " + nama;
    }

}
```

Binatang.java

```
/**
 *
 * @author rama
 * 2211102208
 */
public abstract class Binatang {
    private final String jenis;
    Binatang(String jenis){
        this.jenis = jenis;
    }
    protected abstract void suara();

    @Override
    public String toString(){
        return "seekor "+ jenis;
    }
}
```

Burung.java

```
/**
 *
```

```
 * @author rama
 * 2211102208
 */
public class Burung extends Binatang {

    private final String nama;

    Burung(String nama) {
        super("Burung");
        this.nama = nama;
    }

    @Override
    public void suara() {
        System.out.println("berkicau");
    }

    @Override
    public String toString() {
        return super.toString() + " " + nama;
    }

}
```

Kambing.java

```
/**
 *
 * @author rama
 * 2211102208
 */
public class Kambing extends Binatang {

    Kambing(String nama) {
        super("Kambing");
        this.nama = nama;
    }

    @Override
    public void suara() {
```

```
        System.out.println("mengembik");
    }

    @Override
    public String toString() {
        return super.toString() + " " + nama;
    }
    private final String nama;
}
```

Kucing.java

```
/**
 *
 * @author rama
 * 2211102208
 */
public class Kucing extends Binatang {

    Kucing(String nama) {
        super("Kucing");
        this.nama = nama;
    }

    @Override
    public void suara() {
        System.out.println("mengeong");
    }

    @Override
    public String toString() {
        return super.toString() + " " + nama;
    }
    private final String nama;
}
```

CobaPolimorpic.java

```
/**
 *
 * @author rama
```

```
 * 2211102208
 */
import java.util.Random;
public class CobaPolimorpic {
   public static void main(String[] args) {
       Binatang[] peliharaanku = {new Burung("Kakak
Tua"),new Kambing("Etawa"), new Anjing("Kintamani"),
new Kucing("Anggora")};

       Binatang kesayangan;
       Random pilihan = new Random();
//memilih secara acak

       kesayangan =
peliharaanku[pilihan.nextInt(peliharaanku.length)];
//mengacak bilangan dari 0 sampai length-1 atau (i-1).
       System.out.println(
       "Binatang Kesayangan anda : "+kesayangan);
System.out.print("Suaranya : ");
       kesayangan.suara();
   }
}
```

Output :

```
run:
Binatang Kesayangan anda : seekor Kucing Anggora
Suaranya : mengeong
BUILD SUCCESSFUL (total time: 0 seconds)
```

Keterangan :

Program java diatas maksudnya adalah penggunaan fungsi to string yang di override di setiap class child, pada kasus ini yang di turunkan adalah fungsi toString(). Dalam program ini juga menggunakan polimorfic dalam pembuatannya di fungsi utamanya, terletak pada pendeklarasian peliharaanku terdapat banyak class class yang digunakan untuk object barunya. Jadi class class childnya di deklarasikan dengan class Binatang juga.

**Guided 2**

Source code

Pegawai.java

```
/**
 *
 * @author rama
 * 2211102208
 */
public abstract class Pegawai {

    private final String  namaPeg;
//konstruktor

    public Pegawai(String nama) {
        namaPeg = nama;
    }
//method (get) untuk mengembalikan nama pegawai

    public String namaPegawai() {
        return namaPeg;
    }
//Method abstrak ini diwariskan ke semua kelas yang
diturunkan dari kelas abstrak ini

    public abstract double income();
}
```

Direktur.java

```
/**
 *
 * @author rama
 * 2211102208
 */
public final class Direktur extends Pegawai {

    private double gajiDirektur;
    private double dividenSaham;
//Konstruktor Kelas Direktur
```

```
    public Direktur(String nama, double gaji, double dividen) {
        super(nama); //Memanggil konstruktor kelas Pegawai
        setGajiDirektur(gaji);
        setDividen(dividen);
    }

    public void setGajiDirektur(double gaji) //Mengeset gaji
direktur
    {
        if (gaji > 0) {
            gajiDirektur = gaji;
        } else {
            gajiDirektur = 0;
        }

    }
//Mengeset hasil pembagian dividen keuntungan saham

    public void setDividen(double dividen) {
        if (dividen > 0) {
            dividenSaham = dividen;
        } else {
            dividenSaham = 0;
        }

    }

    public String nama()//Method yang mengembalikan nama
    {
        return super.namaPegawai();
    }

    public String jabatan()//Method yang mengembalikan
jabatan
    {
        return "Direktur";
    }
//Method yang mengembalikan besar gaji direktur
```

```
    public double gajiPerBulan() {
        return gajiDirektur;
    }
//Method yang mengembalikan besar dividen saham

    public double labaDividen() {
        return dividenSaham;
    }
//Pengimplementasian / Pendefinisian method abstract dari
kelas Pegawai
//Method ini mengembalikan besar gaji direktur

    @Override
    public double income() {
        return (gajiDirektur + dividenSaham);
    }
}
```

Test.java

```
/**
 *
 * @author rama
 * 2211102208
 */
import java.text.DecimalFormat;

public class Test {
    public static void main(String args[]) {
        Pegawai pgw;
//Membuat objek referensi dari kelas abstrak //Pegawai
        String output = "";
        Direktur d = new Direktur("Wahyu", 12000000.00,
7500000.00);
        DecimalFormat digitPresisi = new
DecimalFormat("0.00");
        pgw = d;
        /*objek referensi dari kelas abstrak pegawai (pgw)
merefer objek
dari kelas Direktur (d) yang diturunkan dari kelas abstrak
```

```
pegawai */
        System.out.println("\nDEMO INHERITANS,
ENKAPSULASI,POLIMORFI");
        System.out.println("-----------------------------------\n");
// Mencetak informasi Direktur ke console
        System.out.println("Nama : " + d.namaPegawai() + "\n"
                + "Jabatan : " + d.jabatan() + "\n" + "Gaji : "
                + digitPresisi.format(d.gajiPerBulan()) + "\n"
                + "Dividen : " +
digitPresisi.format(d.labaDividen()) + "\n"
                + "Total : " + digitPresisi.format(d.income()) +
"\n");
        System.exit(0);
    }
}
```

Output :

```
run:

DEMO INHERITANS, ENKAPSULASI,POLIMORFI
-----------------------------------

Nama : Wahyu
Jabatan : Direktur
Gaji : 12000000.00
Dividen : 7500000.00
Total : 19500000.00
```

Keterangan :

Pada program diatas sama saja seperti program guided pertama namun yang mencolok adalah ada penginisialisasian object pgw=d, Dimana ‘d’ adalah direktur. Hal tersebut menjadikan method pada class pegawai dan method pada class direktur dapat digunakan di object yang sama.

**Guided 3**

Source code

EkspresiWajah.java

```
/**
 *
```

```
 * @author rama
 * 2211102208
 */
class EkspresiWajah {

    public String respons() {
        return ("Lihat Wajahku ini");
    }
}
```

Gembira.java

```
/**
 *
 * @author rama
 * 2211102208
 */
class Gembira extends EkspresiWajah {

    public String respons() {
        return ("Ha..ha..saya lagi senang =)");
    }
}
```

Sedih.java

```
/**
 *
 * @author rama
 * 2211102208
 */
class Sedih extends EkspresiWajah {

    public String respons() {
        return ("Hiks..hiks.. =(");
    }
}
```

Ekspresi.java

```
/**
```

```
 *
 * @author rama
 * 2211102208
 */
public class Ekspresi {

    public static void main(String args[]) {
        System.out.println("DEMO POLIMORFISME");

System.out.println("=========================");
        EkspresiWajah objEkspresi = new EkspresiWajah();
        Gembira objGembira = new Gembira();
        Sedih objSedih = new Sedih();
        EkspresiWajah[] ekspresi = new EkspresiWajah[3];
        ekspresi[0] = objEkspresi;
        ekspresi[1] = objGembira;
        ekspresi[2] = objSedih;
        System.out.println("Ekspresi[0]:" +
ekspresi[0].respons());
        System.out.println("Ekspresi[1]:" +
ekspresi[1].respons());
        System.out.println("Ekspresi[2]:" +
ekspresi[2].respons());
    }
}
```

Output :

```
run:
DEMO POLIMORFISME
=========================
Ekspresi[0]:Lihat Wajahku ini
Ekspresi[1]:Ha..ha..saya lagi senang =)
Ekspresi[2]:Hiks..hiks.. =(
BUILD SUCCESSFUL (total time: 0 seconds)
```

Keterangan :

Pada program diatas sama saja dengan program guided 1 dan 2 hanya bedanya saat pemmanggilan setiap objectnya kini menggunakan array. Pada program tersebut class EkspresiWajah.java menjadi class parent.

## III. UNGUIDED

### Unguided 1

**Employee.java**

```
/**
 *
 * @author rama
 * 2211102208
 */
public abstract class Employee {
    private String name;
    private String nip;

    public Employee(String name, String nip) {
        this.name = name;
        this.nip = nip;
    }

    public abstract double calculateSalary();

    public void displayInfo() {
        System.out.println("Nama: " + name);
        System.out.println("NIP: " + nip);
        System.out.println("Gaji: " + calculateSalary());
    }
}
```

**SalariedEmployee.java**

```
/**
 *
 * @author rama
 * 2211102208
 */
public class SalariedEmployee extends Employee {
    private double weeklySalary;

    public SalariedEmployee(String name, String nip, double
weeklySalary) {
        super(name, nip);
        this.weeklySalary = weeklySalary;
```

```
    }

    @Override
    public double calculateSalary() {
        return weeklySalary;
    }

    @Override
    public void displayInfo() {
        System.out.println("Salaried Employee");
        super.displayInfo();
        System.out.println("Gaji Perminggu: " + weeklySalary);
    }
}
```

**CommisionEmployee.java**

```
/**
 *
 * @author rama
 * 2211102208
 */
public class CommissionEmployee extends Employee {
    private double baseSalary;
    private double commissionRate;
    private double totalSales;

    public CommissionEmployee(String name, String nip,
double baseSalary, double commissionRate, double
totalSales) {
        super(name, nip);
        this.baseSalary = baseSalary;
        this.commissionRate = commissionRate;
        this.totalSales = totalSales;
    }

    @Override
    public double calculateSalary() {
        return baseSalary + (commissionRate * totalSales);
    }
```

```
    @Override
    public void displayInfo() {
        System.out.println("Commission Employee");
        super.displayInfo();
        System.out.println("Gaji Pokok: " + baseSalary);
        System.out.println("Komisi: " + commissionRate);
        System.out.println("Total Penjualan: " + totalSales);
    }
}
```

**ProjectPlanner.java**

```
/**
 *
 * @author rama
 * 2211102208
 */
public class ProjectPlanner extends Employee {
    private double baseSalary;
    private double commissionRate;
    private double totalProjectResult;

    public ProjectPlanner(String name, String nip, double
baseSalary, double commissionRate, double
totalProjectResult) {
        super(name, nip);
        this.baseSalary = baseSalary;
        this.commissionRate = commissionRate;
        this.totalProjectResult = totalProjectResult;
    }

    @Override
    public double calculateSalary() {
        double tax = 0.05 * baseSalary;
        return baseSalary + (commissionRate *
totalProjectResult) - tax;
    }

    @Override
```

```
    public void displayInfo() {
        System.out.println("Project Planner");
        super.displayInfo();
        System.out.println("Gaji Pokok: " + baseSalary);
        System.out.println("Komisi: " + commissionRate);
        System.out.println("Total Hasil Proyek: " +
totalProjectResult);
    }
}
```

**Main.java**

```
/**
 *
 * @author rama
 * 2211102208
 */
public class Main {
    public static void main(String[] args) {
        SalariedEmployee salariedEmployee = new
SalariedEmployee("Otong", "123", 1000);
        CommissionEmployee commissionEmployee = new
CommissionEmployee("Ucup", "456", 500, 0.1, 2000);
        ProjectPlanner projectPlanner = new
ProjectPlanner("Mawl", "789", 800, 0.05, 3000);

        Employee[] employees = {salariedEmployee,
commissionEmployee, projectPlanner};

        for (Employee employee : employees) {
            employee.displayInfo();
            System.out.println();
        }
    }
}
```

**Output:**

```
run:
Salaried Employee
Nama: Otong
NIP: 123
Gaji: 1000.0
Gaji Perminggu: 1000.0

Commission Employee
Nama: Ucup
NIP: 456
Gaji: 700.0
Gaji Pokok: 500.0
Komisi: 0.1
Total Penjualan: 2000.0

Project Planner
Nama: Mawl
NIP: 789
Gaji: 910.0
Gaji Pokok: 800.0
Komisi: 0.05
Total Hasil Proyek: 3000.0

BUILD SUCCESSFUL (total time: 0 seconds)
```

**Keterangan:**

Program diatas sebenarnya sama saja dengan guided 1,2,3 merupakan pengimplementasian dari polimorpic yaitu lebih dari satu bentuk. Class Employee merupakan class parent dari semua employe yang di buat objeknya di fungsi utama dengan sifat polimorpic.